Pre-Orders for Fault Tolerance

Padmanabhan Krishnan[1]

Technical Report COSC 06/92

A reformatted version of the report is to appear as a paper at the *Sixteenth Australian Computer Science Conference* Citations should refer to the proceedings.

Abstract

We describe a process algebraic approach to the semantics of robust systems. We extend a subset of CCS [14] with multi-set prefixes to model systems with replicated synchronous majority voting. Based on an operational semantics, we define pre-orders which introduces a hierarchy of faulty processes and fault-tolerant processes. We then show how a similar ordering on modal-μ formulae [18] can characterise the fault pre-orders.

[1]Department of Computer Science
University of Canterbury, Private Bag 4800
Christchurch, New Zealand
Email: paddy@cosc.canterbury.ac.nz

1 Introduction

The main characteristic of fault-tolerant (or robust) systems is the ability to cope with errors in software and hardware. Robust systems are usually able to operate correctly in non-ideal environments. For example, the failure of a single processor will not cripple a robust system, the task assigned to the failed processor will be completed. Robust systems are usually reactive systems [17] (i.e., respond to changes in the environment) with the distinguishing feature that the environment can deviate from its 'expected' behaviour without affecting the correctness of the system.

The aim of this paper is to describe a framework in which faulty and fault-tolerant systems can be studied. Cristian [5] describes the various issues in fault-tolerant computation. The first issue is the definition of fault or failure classification and in communicating systems it includes: *omission* fault or failure to send a message, *addition* fault or generation of an spurious message, *value* fault or sending the wrong value, *state-transition* fault or responding incorrectly to the environment and *crash* failure or the inability to interact with its environment.

Associated with a system is a failure model, which is a specification indicating the corrective action on the occurrence of a fault. The failure model chosen for a particular system depends on its functionality. For example, in a student lab environment, shutting down the lab due to an erroneous file-server would be acceptable while a heart-lung machine should not be shut down if a sensor is faulty. Also associated with a fault model is containment, i.e., how to limit the effect of a fault. For example, if backups are available one may shut down a server and activate a backup. If this is done transparently the system as a whole continues to work. Furthermore, these models make assumptions about the ability to identify a fault also called fault detection.

As there are a large number of techniques to detect faults and to recover from them, it is difficult to address all issues in one paper. In this paper we consider the failure classifications of omission, value and addition in replicated systems with synchronous majority voting [6]. As these systems operate in parallel, we develop a theory for replicated systems in the context of theories of concurrent systems.

Process calculi such as ACP [3], CCS [14] and CSP [8] are important formalisms in the description of concurrent systems. Koutny et. al. [9] have developed a trace semantics with extractor functions for replicated CSP processes. However they do not consider explicit fault modelling. Krishnan and McKenzie [10] present a calculus for replicated systems and its bisimulation semantics. They also introduced modelling fault occurrences using action refinements as defined in [2]. However the definitions in [10] were too restrictive. The main reason was the presence of an explicit replication operator, due to which the bisimulation relation was not a congruence.

In this paper we define a calculus based on CCS for replicated systems with a notion of fault injection. Semantic characterisations of the failure classification using pre-orders is defined. The pre-order is relativised with respect to the correct behaviour and if P is less than Q in the pre-order, Q is no more faulty (with respect to the correctness criteria) than P. We also develop a logical characterisation of the pre-orders using the modal-μ calculus [18]. The main achievement of this paper is the development of pre-orders which are relevant to robust systems and their modal characterisation.

2 Replicated Systems

As in CCS [14] we assume a set of atomic actions Λ with typical elements represented by μ_1, μ_2 etc. As we are considering replicated systems with synchronous majority voting, we use multi-sets to indicate the votes for each action. This is justified as in synchronous voting all votes arrive at the 'same' logical time. That is, given a system which is composed of replicated sub-systems, we can assume that the system cannot exhibit an action until each sub-system has voted for the action of its choice.

Definition: 1 *Let $\mathcal{N}$ be the set of natural numbers. A multiset ξ over Λ is a function from Λ to $\mathcal{N}$.*

We represent the multiplicity of elements by superscripts but when the multiplicity is equal to 1 the superscript is not written and actions receiving no vote are not explicitly written. For example, $\mathrm{m} = \{\mu_1^3, \mu_2^2, \mu_3\}$ represents a multiset m where $\mathrm{m}(\mu_1) = 3$, $\mathrm{m}(\mu_2) = 2$, $\mathrm{m}(\mu_3) = 1$ and for all other actions μ $\mathrm{m}(\mu) = 0$.

As we are interested in majority voting, we define an extraction function which identifies the actions that have received maximum number of votes.

Definition: 2 *Define a function* Voted_Action $(\xi) = \{ \mu \mid \forall \mu_1 \in \Lambda, \xi(\mu) \geq \xi(\mu_1)\}$

We are interested in "good" environments, i.e., environments where all votes are identical. In such an environment, all the voting sub-systems reach a consensus on the action to be exhibited.

Definition: 3 *A multiset ξ is said to be perfect iff $\exists\ \mu_1 \in \Lambda$ such that $\xi(\mu_1) > 0$ and $\forall\ \mu_2 \in (\Lambda\text{-}\{\mu_1\})\ \xi(\mu_2) = 0$.*

The syntax for the set of processes is defined as follows.

$$\mathrm{P} ::= nil \mid \xi{\cdot}\mathrm{P} \mid (\mathrm{P} + \mathrm{P}) \mid (\mathrm{P} \mid \mathrm{P})$$

nil represents the terminated process and can exhibit no action. If ξ is 'non-trivial', i.e., at least one action has a non-zero vote, $\xi{\cdot}$P represents the process which has accumulated votes (indicated by ξ) and will exhibit a voted action and then behave as P. For completeness sake one needs to consider the behaviour of empty multisets. While we do not expect a correct specification to involve empty prefixes, an omission fault could erase the prefix. For example, the process ($\mu{\cdot}$P) with an μ-omission fault will behave as P. As shall be seen later ($\mu{\cdot}$P) under an μ-omission fault will be transformed into ($\emptyset{\cdot}$P). The behaviour of ($\emptyset{\cdot}$P) is identical to the behaviour of P. The process (P + Q) represents non-deterministic choice between P and Q, while the process (P | Q) represents parallel combination of P and Q. In this paper we do not consider synchronisation, and as in CCS we permit only one process to evolve in one step.

This syntax is simpler than the one described in the Krishnan and McKenzie [10] model where an explicit replication combinator with the usual action prefix was used. Instead of having an explicit replication combinator we permit a multiset prefix (instead of the usual action prefix). This simplifies the operational semantics considerably. However, the expressive power of the simplified language is not reduced. It is possible to recode a process involving the replication combinator into one using the multi-set prefix. For example, ($\mu_1{\cdot}$P $\amalg$ $\mu_2{\cdot}$Q $\amalg$ $\mu_3{\cdot}$R) can be transformed into $\{\mu_1, \mu_2, \mu_3\}$ ·S, where S is the translation of (P $\amalg$ Q $\amalg$ R).

In the translation, we do not 'discard' a process if its vote was not part of the majority chosen and thus is a relaxation of the model with explicit replication [10]. In [10] for example, the process $(\ (\mu_1 \cdot P) \amalg\ (\mu_2 \cdot Q) \amalg\ (\mu_1 \cdot R))$ would exhibit μ_1 and then evolve to $(P \amalg R)$, i.e., the process Q is discarded. The above process translated to the current setting would exhibit μ_1 but Q will continue to contribute to subsequent behaviour. This implies that in the new model, processes can 'withstand' more faults as the replication factor is not reduced. This is justified as we do not consider Byzantine faults [11] and hence there is no need to eliminate a process from the operating environment.

However, given a process in the original language, it is possible to construct a process with multi-set prefixes whose behaviour is identical to the given process. The process $(\ (\mu_1 \cdot P) \amalg\ (\mu_2 \cdot Q) \amalg\ (\mu_1 \cdot R))$ would be coded as $(\{\mu_1^2, \mu_2\}{\cdot}$S), where S is the appropriate coding of (P $\amalg$ R).

An operational semantics based on labelled transition systems [16] for the above language is defined in figure 1.

Just as we defined a perfect multiset, we define a perfect process as a process which receives a perfect vote for all its actions.

Definition: 4 *A process P said to be perfect if in all sub-terms of the form $\xi{\cdot}P_1$, ξ is perfect.*

Based on the operational (or one step derivation) relation $\longrightarrow$, we use the following abbreviations.

Definition: 5 *a) $P \stackrel{\mu}{\not\longrightarrow}$ iff $\neg\exists\ P'$, such that $P \stackrel{\mu}{\longrightarrow} P'$ b) $P \stackrel{\mu}{\longrightarrow}$ iff $\exists\ P'$ such that $P \stackrel{\mu}{\longrightarrow} P'$*

$P \stackrel{\mu}{\not\longrightarrow}$ indicates that P cannot make a μ move, while $P \stackrel{\mu}{\longrightarrow}$ indicates that P can exhibit μ.

Given a replicated process we now describe our model of fault introduction. Fault introduction is defined as follows

Definition: 6 *Define $(P \dagger \varrho)$ as follows:*

$$nil \dagger \varrho = nil,\ (\xi{\cdot}P) \dagger \varrho = \varrho(\xi){\cdot}P,\ (P + Q) \dagger \varrho = (P \dagger \varrho) + (Q \dagger \varrho)$$

Prefix_1	$\dfrac{\mu \in \text{Voted\_Action}\ (\xi)}{\xi{\cdot}\mathrm{P} \xrightarrow{\mu} \mathrm{P}}$
Prefix_2	$\dfrac{\forall\ \mu{:}\ \xi(\mu) = 0 \quad \mathrm{P} \xrightarrow{\mu} \mathrm{P}'}{(\xi{\cdot}\mathrm{P}) \xrightarrow{\mu} \mathrm{P}'}$
Non-Determinism	$\dfrac{\mathrm{P} \xrightarrow{\mu} \mathrm{P}'}{\begin{array}{c}(\mathrm{P} + \mathrm{Q}) \xrightarrow{\mu} \mathrm{P}' \\ (\mathrm{Q} + \mathrm{P}) \xrightarrow{\mu} \mathrm{P}'\end{array}}$
Parallel	$\dfrac{\mathrm{P} \xrightarrow{\mu} \mathrm{P}'}{\begin{array}{c}(\mathrm{P} \mid \mathrm{Q}) \xrightarrow{\mu} (\mathrm{P}' \mid \mathrm{Q}) \\ (\mathrm{Q} \mid \mathrm{P}) \xrightarrow{\mu} (\mathrm{Q} \mid \mathrm{P}')\end{array}}$

Figure 1: Operational Semantics

Intuitively, if a process has terminated, no fault can affect it, while if a process can perform an action, an occurrence of a fault could alter the action. The presence of non-determinism does not reduce the effect of the fault. The exact definition of $\varrho(\xi)$ will depend on the nature of ϱ and will be discussed later. In general, it is possible to represent a fault as an action refinement function [10]. In this paper we do not specify the refinement function explicitly; rather we define faults as operating on multi-sets directly.

The above definition of fault introduction affects only the first action a process can exhibit and hence models the occurrence of a single failure. This is in keeping with the philosophy of modelling faults as special operations that can occur asynchronously [4].

The idea of approximations as a frame work for verifying satisfaction of specifications by implementations is well known [15]. These approximations can be in the form of a pre-order where ($\mathrm{P} \sqsubseteq \mathrm{Q}$) means that any move P makes can be matched by Q. Therefore, if P is an implementation and Q is a specification, ($\mathrm{P} \sqsubseteq \mathrm{Q}$) requires that all behaviours of an implementation are valid given the specification.

Observational pre-orders (like trace and testing [7]) have been defined for process calculi. In general, for processes P and Q, ($\mathrm{P} \prec \mathrm{Q}$) implies that every behaviour of P can be matched by Q. For example, let P be $(\mu_1 \cdot \mu_2 \cdot \mu_3 \cdot nil + \mu_1 \cdot \mu_2 \cdot \mu_4 \cdot nil)$ and Q be $\mu_1{\cdot}(\mu_2 \cdot \mu_3 \cdot nil + \mu_2 \cdot \mu_3 \cdot nil)$. P is less than Q in the trace pre-order as the traces of P is included in the traces of Q. Similarly, P is less than Q in the testing pre-order as every test (i.e., reacting to external stimuli [7]) that P passes, Q can also pass. Depending on the notion of behaviour different pre-orders can be obtained. For example, $(\mu_1 \cdot \mu_2 \cdot nil + \mu_1 \cdot \mu_3 \cdot nil)$ is trace related but not testing related to $\mu_1{\cdot}(\mu_2 \cdot nil + \mu_3 \cdot nil)$.

Both the trace and testing pre-orders are based on the observable behaviour of a process. However, pre-orders based solely on observable behaviour are not directly useful in the fault-tolerant setting. If $\mathrm{P} \prec \mathrm{Q}$ is to mean that Q can withstand at-least as many faults as P, then the processes with the faults cannot be related based only on observations. If P is μ and Q is μ^3, under a value-altering fault of μ to μ_1 P can exhibit μ_1 which Q cannot match. Therefore, P affected by a fault is not observationally related to Q affected by a fault.

As the behaviour of a faulty process can be significantly different from its behaviour in the absence of faults, a 'correctness' condition is necessary. The correctness criterion distinguishes faulty behaviour from non-faulty behaviour. When relating two processes only the correct behaviour needs to be matched. This indicates the need for an indexed relation. Larsen [12] introduces the idea of equivalences induced by contexts called relativised bisimulation. For example, $\mathrm{P} \prec_C \mathrm{Q}$ relates the behaviours of P and Q in the context C. We use this idea with a different interpretation in developing the fault pre-orders. The preorders we consider do not directly deal with

fault-tolerance. They characterise faulty systems, i.e., where faults are already introduced.

3 Fault Pre-orders

In this section we develop the fault pre-orders in detail. Each type of fault induces a different pre-order. This is natural, as the behaviour of a system with omission failures will not be identical to a system with addition failures. As indicated earlier, an observational pre-order is not sufficient. We define indexed pre-orders of the form P $\prec_C$ Q where C represents the "correct non-faulty" behaviour. The intuitive interpretation is that if P can make a move which C cannot match, one can assume that it is due to the occurrence of a fault. If the fault was of the omission kind, one can operationally assume that P has jumped ahead, while if the fault was of the addition kind, one can assume that P needs to be stepped to reach the same state as C.

3.1 Omission Faults

Definition 7 defines the pre-order induced by omission failure.

Definition: 7 *($P \prec^O_C Q$) iff $C \xrightarrow{\mu} C'$ then*

If $P \xrightarrow{\mu} P'$ then $\exists\ Q'$ such that (($Q \xrightarrow{\mu} Q'$) and ($P' \prec^O_{C'} Q'$))

If $P \not\xrightarrow{\mu}$ and $Q \xrightarrow{\mu}$ then $\exists\ Q'$ such that (($Q \xrightarrow{\mu} Q'$) and ($P \prec^O_{C'} Q'$))

If $P \not\xrightarrow{\mu}$ and $Q \not\xrightarrow{\mu}$ then ($P \prec^O_{C'} Q$)

We use C as the driving agent. If C an perform an action and P can match it, then Q must be able to. This ensures that if P is not faulty then so is Q. If P cannot match the move (due to omission failure) then Q may (no fault) or may not (omission fault) be able to match it. If P (and/or Q) cannot match the move, they are 'held stationary' and C exhibits an action; thus formalising the intuition behind omission faults.

Proposition 1 *$\prec^O_C$ is a pre-order i.e., is reflexive and transitive, ($P \prec^O_{nil} Q$) and ($nil \prec^O_C Q$)*

The *nil* process can be perceived as the result of a process which has been erased by a large number of omission faults and hence is a least element in the pre-order. (P $\prec^O_{nil}$ Q) is valid as the pre-order is indexed by the correctness condition and we constrain the behaviour to a pattern dictated by it. As *nil* can exhibit no action, any two processes are related by $\prec^O_{nil}$. In general, if (P $\prec^O_C$ Q), both P and Q could have unrelated extraneous behaviours as shown by the following example. Let C be $\mu_1 \cdot nil$, P be $(\mu_2 \cdot nil \mid \mu_3 \cdot nil)$ and Q be $(\mu_1 \cdot nil \mid \mu_4 \cdot nil)$. (P $\prec^O_C$ Q) as for the μ_1 move of C, P could exhibit μ_2 and Q could exhibit μ_1. As C evolves to *nil*, $(\mu_3 \cdot nil) \prec^O_{nil} (\mu_4 \cdot nil)$. If the above relation is not desired, a generalisation based on modal transition systems [13] can be used. This will be reported elsewhere at a later date.

The definition of P $\prec^O_C$ Q assumes that faults have been introduced into P and Q and does not require that P is no more *fault-tolerant* than Q. It only indicates that Q is no more *faulty* than P. To define fault-tolerance, we need to define fault introduction and hence need to define $\varrho(\xi)$.

We consider a ϱ to be an μ_1-omission refinement, if it erases μ_1, i.e., $\varrho(\mu_1) = nil$ without affecting any the other action.

Definition: 8 *Let ξ be a multiset and ϱ be an μ_1 omission refinement. $\varrho(\xi) = \xi'$ where*

$\xi'(\mu) = \xi(\mu)$ *if* $\mu \neq \mu_1$ *and*

$\xi'(\mu_1) = (\xi(\mu_1) \dot{-} 1)$ *where $\dot{-}$ is the monus operation.*

Only the votes obtained by the action μ_1 is affected by an μ_1-omission fault introduction. The fault-introduction reduces by one the number of votes received by μ_1.

Proposition 2 *Assume that ξ_p and ξ_q are perfect and ϱ any omission refinement.*

If $\xi_p \cdot P \xrightarrow{\mu} P$, and ($(\xi_p \cdot P) \dagger \varrho$) $\prec^O_{\xi_p \cdot P}$ ($(\xi_q \cdot Q) \dagger \varrho$), then $\xi_q(\mu) \geq 2$ or $\xi_p(\mu) \leq \xi_q(\mu)$ and vice-versa.

The above proposition reiterates the fact that a single replication (or two units) is sufficient to withstand a single instantaneous omission fault. The reason we consider only perfect votes is that if we consider a somewhat erroneous system, an omission fault could manifest itself as other faults. For example, $\langle \mu_1^3, \mu_2^2 \rangle$ with a single μ_1 omission fault can exhibit μ_2 which will then be confused with a μ_1 to μ_2 value fault.

3.2 Value Faults

Definition 9 defines the pre-order induced by value (also called garbling) faults; i.e., faults which alter μ to μ_1.

Definition: 9 *($P \prec_C^V Q$) iff $C \xrightarrow{\mu} C'$ then*

If $P \xrightarrow{\mu} P'$ then $\exists\ Q'$ such that (($Q \xrightarrow{\mu} Q'$) and ($P' \prec_{C'}^V Q'$))

If $P \not\xrightarrow{\mu}$ and $Q \xrightarrow{\mu} Q'$ then $\exists\ \mu_1$, P' such that (($P \xrightarrow{\mu_1} P'$) and ($Q \xrightarrow{\mu} Q'$) and ($P' \prec_{C'}^V Q'$))

If $P \not\xrightarrow{\mu}$ and $Q \not\xrightarrow{\mu}$ then $\exists\ \mu_1$ such that (($P \xrightarrow{\mu_1} P'$) and ($Q \xrightarrow{\mu_1} Q'$) and ($P' \prec_{C'}^V Q'$))

The main difference between definition 7 and definition 9 is that if a matching action cannot be exhibited, a different action *needs* to be exhibited. Furthermore, if P $\prec_C^V$ Q and if both P and Q are faulty (i.e., cannot exhibit the correct action), they are required to exhibit *identical* faulty actions i.e., have identical fault behaviour.

Proposition 3 *$\prec_C^V$ is a pre-order, ($P \prec_{nil}^V Q$) and ($nil \prec_Q^V Q$)*

Note that it is not the case ($nil \prec_C^V$ Q) in general, as if C $\xrightarrow{\mu}$ and Q $\not\xrightarrow{\mu}$, both *nil* and Q are required to make a move. This is because we have indexed the pre-order by C. Alternatively a definition using P as the index or by adding ($nil \prec_C^V$) to it can be considered. The advantages of such a definition needs further investigation.

As in the omission case, we have to define fault injection, for which $\varrho(\xi)$ has to be defined. We consider a refinement function ϱ to be a μ_1-μ_2 value fault, if it alters μ_1 to μ_2, i.e., $\varrho(\mu_1) = \mu_2$, while having no effect on other actions.

The introduction of a fault to a system is defined below.

Definition: 10 *Let ξ be a multi-set and ϱ a μ_1-μ_2 value fault. Define $\varrho(\xi) = \xi'$ where*

$$\xi'(\mu) = \begin{cases} \xi(\mu_1) \dot{-} 1 & \mu = \mu_1 \\ \xi(\mu_2) & \mu = \mu_2 \text{ and } \xi(\mu_1) = 0 \\ \xi(\mu_2) + 1 & \mu = \mu_2 \text{ and } \xi(\mu_1) > 0 \\ \xi(\mu) & \text{otherwise} \end{cases}$$

The above definition consider a μ_1-μ_2 fault and increments the vote of μ_2 only if μ_1 received a positive vote. If μ_1 received no votes, it is not possible to garble it.

Proposition 4 *Let $\xi_p \cdot P \xrightarrow{\mu_1} P$ and $\forall\ \mu \in \Lambda - \{\mu_1, \mu_2\}\ \xi_p(\mu) = \xi_q(\mu) = 0$.*
Let ϱ be a μ_1-μ_2 value fault.
If ($(\xi_p \cdot P) \dagger \varrho$) $\prec_{\xi_p \cdot P}^V$ ($(\xi_q \cdot Q) \dagger \varrho$), then $\xi_p(\mu_1) - \xi_p(\mu_2) \leq \xi_q(\mu_1) - \xi_q(\mu_2)$.

The above proposition indicates if Q is at least as fault-tolerant as P, Q′s difference in votes for μ_1 and μ_2 is no less than P′s difference in votes.

The converse of the above proposition is also true. The converse proposition will not hold if ξ_p or ξ_q had 'significant' votes for other actions. For example, $\langle \mu_1^3, \mu_2^1, \mu_3^2 \rangle$ under ϱ can exhibit μ_1, while $\langle \mu_1^4, \mu_3^4 \rangle$ under ϱ cannot. Though the value fault changed μ_1 to μ_2 the presence of μ_3 changes the nature of the fault.

3.3 Addition Faults

The treatment of addition faults is different from the treatment of omission and value faults. In communicating systems, an addition fault adds a message to the system. As we are considering a frame-work with votes, the issue of *when* the additional message arrives is crucial. It is possible to assume no bound on when the additional message could arrive. Under such an assumption, the theory becomes unmanageable. In this paper we consider an 'atomic' semantics; i.e., assume that the additional message arrives along with the actual message.

Intuitively, ϱ models an μ addition fault if it increments the number of votes received by μ by one while not affecting the other actions. We define the effect of an addition refinement on the current state of votes as follows.

Definition: 11 *A refinement ϱ is called an μ addition fault if*

$$\forall\ \xi,\ \varrho(\xi) = \xi' \text{ where } \xi'(\mu) = \xi(\mu) + 1 \text{ and } \forall\ (\mu_1 \neq \mu),\ \xi'(\mu_1) = \xi(\mu_1).$$

The above definition ensures that the 'correct' action and the 'faulty' additional action are considered by the voting mechanism *simultaneously*.

Fault introduction via addition refinement will be observationally similar to value-fault as one action can be altered to another. If a system exhibits μ_1 instead of the expected μ an observer cannot determine if the fault was due to garbling or addition. Hence the definition of the pre-order induced by addition-faults is identical to definition 9.

Definition 12 describes the pre-order induced by addition faults and is presented only for the sake of completeness.

Definition: 12 *$(P \prec^A_C Q)$ iff $C \xrightarrow{\mu} C'$ then*

If $P \xrightarrow{\mu} P'$ then $\exists\ Q'$ such that, ($(Q \xrightarrow{\mu} Q')$ and $(P' \prec^A_{C'} Q')$)

If $P \not\xrightarrow{\mu}$ and $Q \xrightarrow{\mu} Q'$ then $\exists\ \mu_1$, P' such that, ($(P \xrightarrow{\mu_1} P')$ and $(P' \prec^A_{C'} Q')$)

If $P \not\xrightarrow{\mu}$ and $Q \not\xrightarrow{\mu}$ then $\exists\ \mu_1$, P', Q' such that, ($(P \xrightarrow{\mu_1} P')$ and $(Q \xrightarrow{\mu_1} Q')$ and $(P' \prec^A_{C'} Q')$)

Proposition 5 *Let there exist a unique μ such that $\xi_p{\cdot}P \xrightarrow{\mu} P'$ and let ϱ be an μ_1 addition fault. If ($(\xi_p{\cdot}P) \dagger \varrho) \prec^V_{\xi_p \cdot P}$ ($(\xi_q{\cdot}Q) \dagger \varrho$), then $(\xi_p(\mu)$ - $\xi_p(\mu_1) < 1)$ or $(\xi_q(\mu)$ - $\xi_q(\mu_1) > 1)$*

The above proposition is similar to proposition 4 except that it requires an unique initial move. The difference is mainly because garbling faults can reduce votes, while addition faults only add to votes.

This concludes the definition of the fault pre-orders. In the next section we present a modal logic characterisation of the omission and value fault pre-orders.

4 Modal Characterisation

It has been shown that the usual bisimulation semantics for process algebras can be characterised by the modal-μ calculus [18]. In this section we characterise certain aspects of fault-tolerance using a subset of the modal-μ logic. The fragment of the modal-μ we use is as follows

$$\varphi ::= \text{True} \mid \langle\mu\rangle\varphi \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1 \vee \varphi_2)$$

Associated with the logical formulae and the set of processes is a satisfaction relation. A process P satisfies a formula $\langle\mu\rangle\varphi'$ (written as $\mathrm{P} \models \langle\mu\rangle\varphi'$) iff there is a P′ such that $\mathrm{P} \xrightarrow{\mu} \mathrm{P}'$ and $(\mathrm{P}' \models \varphi')$. All processes satisfy True while $\wedge$ and $\vee$ represent logical conjunction and logical disjunction respectively.

In the following two sections we show how omission and garbling faults and the associated fault pre-orders can be logically described. As addition fault is similar to value fault, we do not consider its logical characterisation.

4.1 Omission Faults

As an omission fault introduction can prevent a process from exhibiting an initial action, the following proposition holds.

Proposition 6 *If* $P \models \langle\mu\rangle\varphi$ *and* ϱ *an* μ*-omision, then* $(P \dagger \varrho) \models \langle\mu\rangle\varphi \vee \varphi$ *and if* ϱ *is an* μ_1 *omission and* $\mu \neq \mu_1$, $(P \dagger \varrho) \models \langle\mu\rangle\varphi$.

Towards the characterisation of the omission fault pre-order ($\prec_C^O$), define a translation function $[\![\varphi]\!]_O$ which identifies the possible formulae that a 'faulty' process can satisfy given that the 'correct' process satisfies φ.

Definition: 13 $[\![True]\!]_O = \{ True\}$

$[\![\langle\mu\rangle\ \varphi]\!]_O = \{\langle\mu\rangle\ \varphi' \mid \varphi' \in [\![\varphi]\!]_O\} \cup [\![\varphi]\!]_O$

$[\![\varphi_1 \oplus \varphi_2\ ]\!] = \{\varphi_i \oplus \varphi_j \mid \varphi_i \in [\![\varphi_1]\!]$ *and* $\varphi_j \in [\![\varphi_2]\!]\}$ *for* $\oplus \in \{\vee, \wedge\}$

The above definition transforms a given formula into ones where omission faults have occurred at arbitrary instances. The translation by itself is not sufficient as if P $\prec_C^O$ Q, it need not be the case that all formulae that P satisfies Q will. For instance, let (C $\models\langle\mu_1\rangle\langle\mu_2\rangle$True) and P satisfy $\langle\mu_2\rangle$True. If Q is less faulty than P, (Q $\not\models \langle\mu_2\rangle$True) but (Q $\models\langle\mu_1\rangle\langle\mu_2\rangle$True). This indicates the need for a hierarchy of formulae which denotes 'less' faulty. As in the pre-order case, the hierarchy has to be indexed by a correctness formula.

Definition: 14 *For every formula* φ*, define an ordering on* $[\![\varphi]\!]_O$ *as follows.*

- $\forall\varphi_1, \varphi_2,\ \varphi_1 \sqsubseteq^O_{True}\ \varphi_2$
- $\forall\ \varphi' \in [\![\langle\mu\rangle\varphi]\!]_O,\ \varphi' \sqsubseteq^O_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi$
- $\forall\ \varphi_1, \varphi_2 \in [\![\varphi]\!]_O\ \varphi_1 \sqsubseteq^O_\varphi\ \varphi_2$ *implies*
  - $\varphi_1 \sqsubseteq^O_{\langle\mu\rangle\varphi}\ \varphi_2$
  - $\varphi_1 \sqsubseteq^O_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi_2$
  - $\langle\mu\rangle\varphi_1 \sqsubseteq^O_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi_2$

As $\sqsubseteq^O_\varphi$ indicates a fault hierarchy and as all processes satisfy True, any two formulae are related if the correctness criteria is True. Every formula in $[\![\varphi]\!]$ represents a formula that a potentially faulty process could satisfy. Hence all elements of $[\![\varphi]\!]$ are less than φ. The third aspect of the definition deals with 'future' faults. If future behaviour indicates that a formula φ_1 is more faulty than another φ_2 under φ, then both could suffer omission faults and hence are related under $\langle\mu\rangle\varphi$ or the formula $\langle\mu\rangle\varphi_2$ is less faulty than both φ_1 and $\langle\mu\rangle\varphi_1$.

Proposition 7 $\sqsubseteq^O_\varphi$ *is a pre-order.*

4.2 Value Fault

A garbling fault can alter the initial action that a process can perform. Unlike an omission fault, a value fault ensures that the modified process can exhibit an action if the original process could.

Proposition 8 *If* $P \models \langle\mu\rangle\varphi$ *and* ϱ *an* μ-μ_1 *value fault, then* $(P \dagger \varrho) \models \langle\mu\rangle\varphi \vee \langle\mu_1\rangle\varphi$ *and if* ϱ *is an* μ_1-μ_2 *value fault and* $\mu \neq \mu_1$, $(P \dagger \varrho) \models \langle\mu\rangle\varphi$.

The above proposition characterises value-fault introduction. As for the omission faults case we define sets of formulae equipped with an ordering which characterises the fault pre-order $\prec_C^V$.

Definition: 15 $[\![True]\!]_V = \{ True\}$

$[\![\langle\mu\rangle\ \varphi]\!]_V = \{\langle\mu_1\rangle\ \varphi' \mid \varphi' \in [\![\varphi]\!]_V$ *and* $\mu_1 \in \Lambda\}$

$[\![\varphi_1 \oplus \varphi_2\ ]\!] = \{\varphi_i \oplus \varphi_j \mid \varphi_i \in [\![\varphi_1]\!]$ *and* $\varphi_j \in [\![\varphi_2]\!]$ *for* $\oplus \in \{\vee, \wedge\}\}$

Definition: 16 *For every formula φ, define an ordering on $[\![\varphi]\!]_V$ as follows.*

- $\forall \varphi_1, \varphi_2,\ \varphi_1 \sqsubseteq^O_{True} \varphi_2$
- $\forall\ \varphi' \in [\![\langle\mu\rangle\varphi]\!]_V\ \ \varphi' \sqsubseteq^V_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi.$
- $\forall \varphi_1, \varphi_2 \in [\![\varphi]\!]_V$, *such that* $\varphi_1 \sqsubseteq^V_\varphi \varphi_2$, *implies*
  - $\forall\ \mu_1 \in \Lambda,\ \langle\mu_1\rangle\varphi_1 \sqsubseteq^V_{\langle\mu\rangle\varphi}\ \langle\mu\rangle\varphi_2$
  - $\langle\mu_1\rangle\varphi_1 \sqsubseteq^V_{\langle\mu\rangle\varphi}\ \langle\mu_1\rangle\varphi_2$

The definition of $\sqsubseteq^V_\varphi$ is similar to that of $\sqsubseteq^O_\varphi$ except in the third case where instead of action omission, we alter μ to μ_1 in φ_1, indicating a garbling fault.

Proposition 9 $\sqsubseteq^V_\varphi$ *is a pre-order.*

The following proposition indicates that the modal formulae equipped with the appropriate ordering captures the fault hierarchy. As Q satisfies a formula 'higher up' in the pre-order, Q is less faulty than P.

Proposition 10 *If $C \models \varphi$, $\forall \varphi' \in [\![\varphi]\!]_X$ (for $X \in \{O, V\}$) $P \models \varphi'$ implies $\exists \varphi'' \in [\![\varphi]\!]_X$ such that $Q \models \varphi''$.*

5 Conclusion and Future Work

In this paper we have presented a simple syntax and operational semantics for replicated systems. We have considered three types of faults and defined pre-orders induced by them. These pre-orders were indexed by a correctness criteria. If P $\prec^T_C$ Q it indicates that Q is no more faulty than P for faults of type T given correctness criteria C. We have also defined fault introduction and presented a few preliminary results relating the fault pre-orders and fault-introduction. We have presented a modal logic characterisation of the fault-introduction and fault pre-orders.

The main issues that need further investigation include applying the technique to other types of fault tolerant systems, considering recursive processes and communication and extending the modal characterisation to the full modal-μ calculus.

Synchronous majority voting is only one technique to attain fault tolerance. As replication of sub-systems can be expensive, other techniques such as resourceful systems [1] are popular. The applicability of this work to such techniques needs further investigation.

In this paper we have not considered recursion. The main issue in fault-tolerant recursive systems is whether subsequent unfoldings are the modified faulty process or the original process. This depends of whether the fault is considered permanent or transient.

The reason for excluding communication is that the effect of a fault on complementary and hidden actions needs to be considered. Consider, for example, the CCS process (say P) $(\mu{\cdot}\mathrm{Q} \mid \overline{\mu}{\cdot}\mathrm{R}) \backslash\{\mu\}$ and its behaviour under an μ-omission fault. If the μ-omission fault also omits $\overline{\mu}$, P is weakly bisimilar to P under fault. However, there is no reason to believe that faults will be 'well-behaved'. If μ-omission does not affect $\overline{\mu}$, then P affect by the fault is related to Q. While this is acceptable, the process $(\mu_1{\cdot}\mathrm{Q} \mid \overline{\mu_1}{\cdot}\mathrm{R}) \backslash\{\mu_1\}$ is not affected by an μ-omission. Therefore, the choice of local names has an impact on the fault semantics. One could argue that a fault should not affect hidden actions but such an assumption would not be realistic.

In the modal characterisation we did not consider explicit negation nor the necessity ($[\mu]$) modality. The reason is that we have been unable to provide transformations that characterise fault-tolerance. For example, consider the formula $[\mu_2]$False and a process (say P) $\mu_1 \cdot \mu_2$. While P satisfies the formula, P under an μ_1 omission does not satisfy the impossibility requirement. In this particular case the faulty process will satisfy $\langle\mu_2\rangle$True. A general scheme to translate and impose an order on modal formulae involving negation is under investigation.

Acknowledgements

The author acknowledges the helpful comments and suggestions given by the anonymous referees. This research has been partially supported by University of Canterbury Grant No 1787123.

References

[1] R. J. Abbot. Resourceful Systems for Fault Tolerance, Reliability and Safety. *ACM Computing Surveys*, 22(1), 1990.

[2] L. Aceto and M. Hennessy. Adding Action Refinement to a Finite Process Algebra. In *ICALP -91, LNCS 510*. Springer Verlag, 1991.

[3] J. A. Bergstra and J. W. Klop. Process Theory Based on Bisimulation Semantics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS 354*, pages 50–122. Springer Verlag, 1988.

[4] F. Cristian. A Rigorous Approach to Fault-Tolerant Programming. *IEEE Transactions on Software Engineering*, 11(1):23–31, 1985.

[5] F. Cristian. Understanding Fault-Tolerant Distributed Systems. *Communications of the ACM*, 34(2):56–78, February 1991.

[6] R. E. Harper and J. H. Lala. Fault-tolerant Parallel Processor. *AIAA Journal of Guidance, Control and Dynamics*, 14(3):554–563, May-June 1991.

[7] M. C. B. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.

[8] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International, 1985.

[9] M. Koutny, L. Mancini, and G. Pappalardo. Formalising Replicated Distributed Processing. In *Proceedings of the 10th Symposium on Reliable Distributed Systems*, pages 108–117, Pisa, Italy, 1991. IEEE.

[10] P. Krishnan and B. J. McKenzie. A Process Algebraic Approach to Fault-Tolerance. In *Proceedings of the 15th Australian Computer Science Conference*, pages 473–485, Hobart, January 1992.

[11] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[12] K. G. Larsen. A Context Dependent Equivalence Between Processes. *Theoretical Computer Science*, 49:185–215, 1987.

[13] K. G. Larsen. Modal specifications. In *Proceedings of the Workshop in Automatic Verification Methods for Finite-State Systems: LNCS 407*, pages 232–246. Springer Verlag, 1989.

[14] R. Milner. *Communication and Concurrency*. Prentice Hall International, 1989.

[15] E. R. Olderog and C. A. R. Hoare. Specification-oriented Semantics for Communicating Processes. In *ICALP -83, LNCS 154*. Springer Verlag, 1983.

[16] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[17] A. Pnueli. Linear and Branching Structures in the Semantics and Logics of Reactive Systems. In *ICALP -85: LNCS 194*, pages 15–32. Springer Verlag, 1985.

[18] C. Stirling. An Introduction to Modal and Temporal Logics for CCS. In *Joint UK/Japan Workshop on Concurrency:LNCS 491*, pages 2–20, 1989.